Efficient Global Multi Parameter Calibration for Complex System Models Using Machine-Learning Surrogates

Julius Aka[1,2] Johannes Brunnemann[1] Svenne Freund[2] Arne Speerforck[2]

[1]XRG Simulation GmbH, {aka,brunnemann}@xrg-simulation.de
[2]Hamburg University of Technology, {julius.aka,svenne.freund,arne.speerforck}@tuhh.de

Abstract

In this work, we address challenges associated with multi parameter calibration of complex system models of high computational expense. We propose to replace the Modelica Model for screening of parameter space by a computational effective Machine-Learning Surrogate, followed by polishing with a gradient-based optimizer coupled to the Modelica Model.
Our results show the advantage of this approach compared to common-used optimization strategies. We can resign on determining initial optimization values while using a small number of Modelica model calls, paving the path towards efficient global optimization. The Machine Learning Surrogate, namely a Physics Enhanced Latent Space Variational Autoencoder (PELS-VAE), is able to capture the impact of most influential parameters on small training sets and delivers sufficiently good starting values to the gradient-based optimizer.
In order to make this paper self-contained, we give a sound overview to the necessary theory, namely Variational Autoencoders and Global Sensitivity Analysis with Sobol Indices.
Keywords: Sensitivity Analysis, Sobol-Indices, Variational Autoencoders, VAE, Physics-Enhanced Latent Space Variational Autoencoder, PELS-VAE, Model Calibration, Global Optimization, Machine Learning Surrogate

1 Introduction

To enable model based investigation of "real world" technical systems the underlying Modelica system models can quickly grow in size and computational expense. When they are applied in extensive parameter studies, in particular for model calibration or model based optimization, computation becomes a resource intensive task: if the objective function cannot be decomposed into submodel dependencies but depends on the model as a 'whole', then also the whole model needs to be simulated.

In practice optimization based on such models is limited to a few varied parameters and to local, gradient based optimization algorithms. If the modeller has sufficient knowledge on the model, reasonable choices of relevant parameters as well as starting points for the local optimization algorithm can be made from experience. But for complex models this empirical approach may suffer from overlooking parameters and the optimization algorithm running into local minima of the objective function due to the chosen starting points in parameter space.

In this paper we address these issues with a combined approach: A Machine Learning Model, namely a Physics Enhanced Latent Space Variational Autoencoder (PELS-VAE) (Martínez-Palomera, Bloom, and Abrahams 2020; Zhang and Mikelsons 2022) is trained on data generated by the Modelica model. It captures the dependences of model output to the most influential parameters, determined by a preceding sensitivity analysis (Sobol 1993), while requiring a limited set of training data. This surrogate is computationally cheap, and can be used to apply a global optimization algorithm that relies on a large number of model runs. After this global screening, a subsequent local optimization based on the original physical model (polishing) is performed.

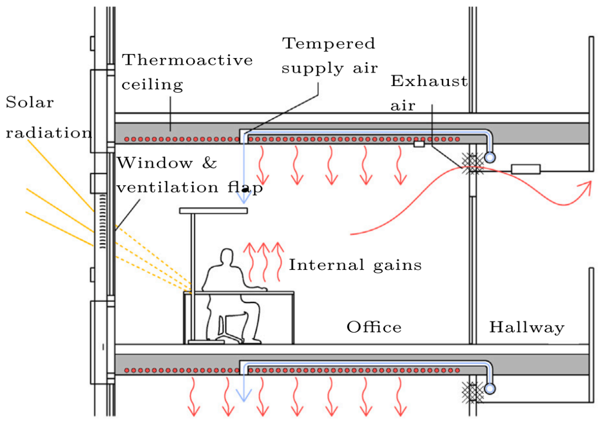


Figure 1. Schematic of standard single office, taken from (Freund and Schmitz 2021)

We choose to test our approach on a computational inexpensive, thermal Modelica model of a single office (Figure 1) with measurement data available for calibration (Freund and Schmitz 2021). Like this, data generation for the machine learning models is fast and we are able to focus on the application of the PELS-

VAE for parameter calibration, while being able to cross check all obtained results against a brute force global optimization based on the original model.

Various Optimization Tools suitable for Modelica models already exist, like the *Dymola Optimization Library*, *GenOpt* (University of California 2023), *ModestPy* for Parameter Estimation with FMUs (Arendt et al. 2018), *AixCaliBuHA* (Wüllhorst et al. 2022) or *ModelOpt* (XRG Simulation GmbH 2023). All of these tools vary in detail, but build on common-known global and local Optimization Algorithms like Particle Swarm Optimization, Genetic Algorithms, Sequential Least Squares or Nelder-Mead Algorithm and do not generate surrogate models.
In contrast to this, surrogate based optimization aims to represent computationally expensive models by the use of a simpler surrogate to significantly save computational resources. Different kinds of surrogate models like linear regression, support vector regression, radial basis functions or kriging (Gaussian process regression) are commonly used (Bhosekar and Ierapetritou 2018). Artificial Neural Network as a generalization of regression models are also a possible surrogate choice. A promising subclass is Bayesian Optimization, which consists of a probabilistic surrogate model and a sequential called loss function that enables optimal, active sampling of the objective function that should be replaced (Shahriari et al. 2016). Bayesian Optimization proved efficient in parameter calibration of a Modelica-modeld HVAC-system (Martinez-Viol et al. 2022). In comparison to these techniques, our approach replaces the actual physical model for a fixed scenario, not the cost function of an optimization objective.

This paper is organised as follows: section 2 introduces the used Modelica model of an office room, the PELS-VAE architecture and training, as well as the applied optimization techniques. In section 3 we present the results of applying our approach for calibration of the Modelica model. Finally we summarize our findings and give an outlook to present and future work in section 4. In Appendix B, we sketch the applied global sensitivity analysis.

2 Methods

2.1 Calibration Problem

The modelled thermal zone is a room of a large-scale office-building ($46\,500\,\mathrm{m}^2$) and high energy efficency (primary energy demand $< 70\,\mathrm{kW\,h\,m}^{-2}$) (Freund and Schmitz 2021). The buildings operation has been explored in previous research projects ((Niemann and Schmitz 2020), (Duus and Schmitz 2021), (Freund and Schmitz 2021)). For example, Model-Predictive-Control (MPC) was used to enhance thermal user-comfort and decrease energy demand (Freund 2023). MPC requires accurate models which can be obtained by calibrating Modelica-Models with measurement data.
A scheme of an office is shown in Figure 1. Heat is supplied by thermal activated ceilings (TAC), i.e. by circulating warm water through pipes in the concrete core of the slabs, and mechanical ventilation with preheated supply air. The large area of the ceilings allows the usage of heat-pumps for low temperature heating, while the high thermal capacity of the concrete slabs enables considerable time delay between heat supply to the ceiling and heat supply to the room. For this building, measurement data is recorded since 2014 at more than 1100 sensors every minute. 32 office spaces are equipped as reference zones with various sensors. (Freund and Schmitz 2021)
For this project, we use the same data than in prior studies (Freund 2023). The calibration target is to fit the model output T_{Air} to the recorded measurement $T_{\mathrm{Air,meas}}$ by adjusting the model parameters $\boldsymbol{\theta}$ within their bounds$[\boldsymbol{\theta}_-, \boldsymbol{\theta}_+]$, employing an error metric such as the Mean Squared Error (MSE):

$$\min_{\boldsymbol{\theta}} \quad \frac{1}{T}\sum (T_{\mathrm{Air}}(t_i) - T_{\mathrm{Air,meas}}(t_i))^2 \qquad (1)$$
$$\text{subject to } \boldsymbol{\theta}_- \leq \boldsymbol{\theta} \leq \boldsymbol{\theta}_+$$

which is in general a constrained, nonlinear optimization problem.
The recorded data consists of several timeseries that serve as an input to the physical model of the thermal zone. The model inputs are outside air temperature T_{A}, supply temperature of the corresponding TAC heating circuit $T_{\mathrm{Sup,TAC}}$, boolean signal of supply $y_{\mathrm{Sup,TAC}}$, supply temperature of mechanical ventilation $T_{\mathrm{Sup,MV}}$, boolean signal of supply $y_{\mathrm{Sup,MV}}$, global solar radiation and occupancy state. For the heat exchange at sun-exposed walls, an equivalent outdoor air temperature $T_{\mathrm{A,Eq}}$ is used. Internal heat gains by persons, lighting or other equipment $\dot{Q}_{\mathrm{Int}}$ are calculated using the by constant heat gain factor multiplied with an heuristic based on measured occupancy state and the buildings electric energy consumption load profile. Internal and external heat gains are split into convective parts acting on the air volume and radiative parts acting on the internal masses. We use data of the identification-timeframe 21.02.2018 - 14.03.2018 (Freund 2023).

2.1.1 Gray-Box Model

In this work, a Gray-Box Model introduced by (Freund and Schmitz 2021) shall be calibrated. Gray-Box Modeling referes to a modeling approach, where a physical model is combined with data-driven approaches. Physical knowledge is used to derive a model structure, while parameters are identified using measurement data (Kathirgamanathan et al. 2021).

The gray-box model (Figure 2) consists of seven resistances and four capacities (R7C4 model). The four

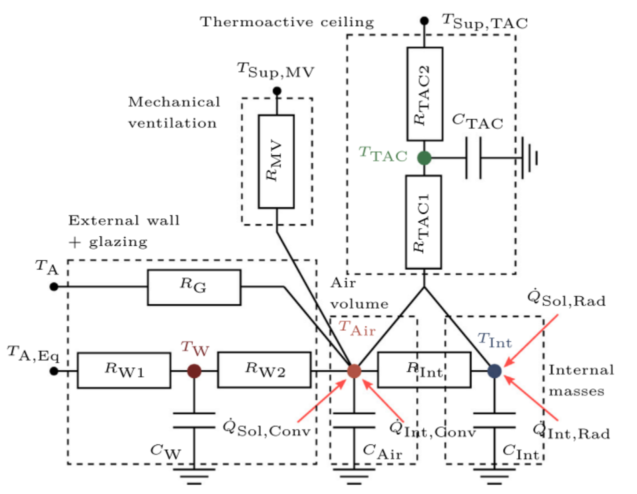


Figure 2. RC network representation of gray-box zone-model with 7 Resistances and 4 Capacitors (R7C4) (Freund and Schmitz 2021).

state variables T_{W} (external wall temperature), T_{Air} (indoor air temperature), T_{Int} (temperature of internal masses) and T_{TAC} (TAC core temperature) correspond to the four thermal capacities $C_{\mathrm{W}}, C_{\mathrm{Air}}, C_{\mathrm{Int}}$ and C_{TAC}.

Based on the EMPA model (Koschenz and Lehman 2000), a simplified model for the TAC is used consisting of two resistances R_{TAC1} and R_{TAC2}. By assuming equal room temperatures below and above the thermoactive ceiling, the two heat flow paths to respectively the room above and below the ceiling can be transformed into a single heat flow path, resulting in a R2C1 TAC model (Sourbron 2012).

The external wall is modeled with two resistances for the envelop (R_{W1} and R_{W2}) and one resistance for the glazing R_{G}. Mechanical ventilation is represented with one resistance R_{MV}. The resistance R_{Int} describes the heat exchange between the air volume and the internal masses. Heat exchange between adjacent zones is neglected, since the heating control is for all zones of a building section the same.

Consequently, the simulation model has 11 parameters (see Table 1). Additionally, we introduce the parameter f_{sol} to tune the fraction of the window projected global radiation flowing to the office and the parameter $\dot{Q}_{\mathrm{Int}}$ as heat gain factor of the heuristic occupancy signal. The initial temperature $T_{\mathrm{TAC}}(t=0)$ of the TAC as the mass with the highest capacity is introduced as a parameter to the optimization problem. Estimated values for these parameters are obtained by using the documentation of constructional elements and values from literature. These estimates are used to generate training data for the autoencoder models, which is for most parameters performed in the range of $\frac{1}{5}$ to 5 times the estimated value. We choose these broad ranges in order to account for situations, where little knowledge on the estimates is available. In practice they should be narrowed as much as possible by available information.

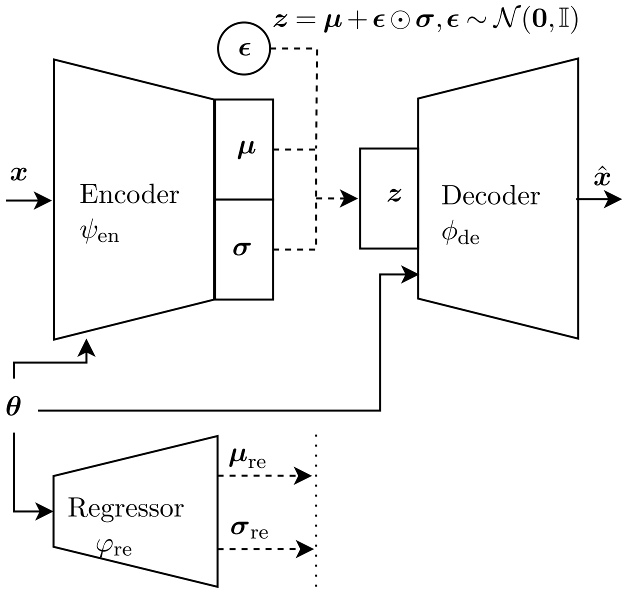
Table 1. Description of the 14 RC-Model Parameters and Corresponding Parameters of the Modelica Model.

RC-Model	Description	Modelica Model
C_{W}	Wall Capacity	`cExt1`
C_{Air}	Air Capacity	`b`
C_{Int}	Internal Masses	`cInt`
C_{TAC}	Thermoactive Ceiling (TAC) Capacity	`cTABS`
R_{TAC1}	TAC Resistance Capacity/Room	`rZone`
R_{TAC2}	TAC Resistance Pipe/-Capacity	`rPipe`
R_{W1}	Wall Resistance Outdoor/Capacity	`rExt1`
R_{W2}	Wall Resistance Capacity/Room	`rExt2`
R_{G}	Window Resistance	`UWin`
R_{MV}	Mechanical Ventilation	`VSup`
R_{Int}	Internal Heat Exchange	`rInt`
f_{sol}	Solar Gain Fraction	`fSol`
$\dot{Q}_{\mathrm{Int}}$	Internal Heat Gains	`qIntOcc`
$T_{\mathrm{TAC}}(t=0)$	Initial Value	`TTABSInit`

The obtained model is exported by using the Functional Mock-up Interface (*FMI*) standard and used in Python-Scripts with *FMPy* (FMPy 2023). We simulate with a time step of 1800 s.

2.2 Physics-Enhanced Latent Space Variational Autoencoder

The general idea of Autoencoders is to encode data of a dataset in a lower-dimensional compression that is sufficient to represent the variation within that dataset. For example, a collection of images of people could be reduced to characteristics like gender, hair color, skin color, pose etc. From this compression, data can be reconstructed with a decoder that learned the influence on the compression of these attribute variations to reconstruct an image from it. In general, the lower-dimensional compression is said to be in a "latent space", i.e. a space whose behavior is hidden and cryptic to us. A Encoder-Decoder Neural Network structure is an unsupervised learning technique. However, the representation of attributes in latent space can be learned, i.e. by a neural network (”Regressor”). By only using the Regressor and the Decoder, new data can be generated, such that an Generative Adversarial Network (GAN) is obtained. A challenge is to chose an adequate dimension for the latent space to prevent the network from just memorizing the data (Jordan 2018a). Various tech-



$z = \mu + \epsilon \odot \sigma, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$
ϵ
x
Encoder ψ_{en}
μ
σ
z
Decoder ϕ_{de}
$\hat{x}$
θ
Regressor φ_{re}
μ_{re}
σ_{re}

Figure 3. Physics-Enhanced Latent Space Variational Autoencoder (PELS-VAE). The time-series $\boldsymbol{x}$ is introduced to the Encoder ψ_{en}, which transforms it to a latent-space distribution with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}$, which can be decoded by the Decoder ϕ_{de} by sampling $\boldsymbol{z}$ with the auxiliary Gaussian variable $\boldsymbol{\epsilon}$ to reconstruct the time-series as $\hat{\boldsymbol{x}} = \frac{1}{L}\sum^{L} \phi_{de}(\boldsymbol{z})$. The Regressor φ_{re} is trained simultanously to predict the mean and variance of the latent space distribution. As in (Martínez-Palomera, Bloom, and Abrahams 2020), the physical parameters $\boldsymbol{\theta}$ are introduced to all models. (Zhang and Mikelsons 2022)

niques have been proposed for this regularization, and a widely used approach is to learn probability distributions within the autoencoder structure, making it an Variational Autoencoder (VAE). Hands-on explanation for Autoencoders can be found in (Jordan 2018a), while for VAE in (Jordan 2018b).
Within this paper, we build on the implementation of (Zhang and Mikelsons 2022) to predict time-series $\boldsymbol{x}$ (i.e. our temperature trajectories), with its architecture shown in Figure 3.

For the interested reader, a more detailed explanation of the theory behind the Autoencoder and its training loss function is provided in Appendix A.

2.3 Training Data Generation

To train the PELS-VAE model to mimic the behaviour of the physical model, i.e. learning the behaviour $\boldsymbol{x}(\boldsymbol{\theta})$, the machine learning model needs to be exposed to labeled training data $(\boldsymbol{x}|\boldsymbol{\theta})$. Therefore, we sample n times uniformly in parameter space:

$$\boldsymbol{\theta} \sim \mathcal{U}(\boldsymbol{\theta}_{-}, \boldsymbol{\theta}_{+}) \tag{2}$$

and run the physical Modelica Model to get the posterior $\boldsymbol{x}$ of $\boldsymbol{\theta}$. As the purpose of this paper is to determine possible reduction in required simulations of the physical model, we generate training sets with different sizes in the range (32 to 4096). Validation and test sets have a size of 320 samples. To make results comparable, validation and test sets are the same for all models. The validation set is used to validate the model during the training process to select well-generalizing models and to early stop the training if no further improvement is happening. The test-set is used to determine the final performance of the model, unbiased by the selection through the test set.
The training data should cover well the parameter space as well as the output space, which can be checked by plotting the corresponding confusion plots (combining every θ_i with each other) and plotting all outputs of the physical model. Combining these plots of the model outputs with available measurement data, allows to make a first check if the designed physical model is able to capture the observed behaviour (see Figure 4).

2.4 Optimization-Based Parameter Identification

This paper aims to calibrate a model by minimizing the Mean Squared Error (MSE) between the model output and recorded measurements to determine a globally minimizing parameter combination. An overview of the applied methods is provided in Table 2 and discussed further below.

To demonstrate the superiority of our proposed method over existing optimization techniques, we combine a *FMU* of the Modelica model with selected optimization methods from *SciPy* and compare them with our introduced methods that use the Physics-Enhanced Latent Space Variational Autoencoder (PELS-VAE).

The investigated methods that combine a *SciPy* optimizer with an *FMU* encompass scalar or vector-like objectives, gradient-descent or non-gradient-descent methods, and can be categorized as either local or global optimization techniques. We anticipate gradient-based optimizers to converge quickly and expect further improvements for the LS-TRF approach, which utilizes residuals as the objective, as the optimizer gains more knowledge about the optimization step consequences compared to scalar objectives.
On the other hand, we consider Differential Evolution, a genetic algorithm (GA), a global optimization technique, albeit with the drawback of requiring a higher number of model evaluations.
All local techniques in this study necessitate initial values for the parameters, which may be challenging to derive in practical applications. To address this, we combine each local technique with a multistart approach, where the optimization is initiated n_{start} times using starting values randomly distributed around the given initial parameter values.

Based on these evaluations, we propose to combine a well-trained computationally cheap PELS-VAE

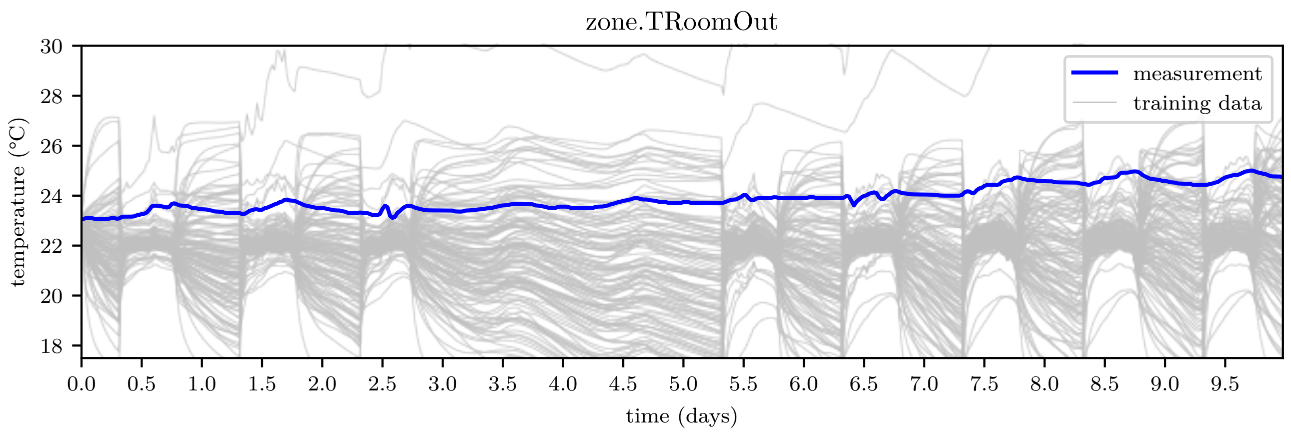


Figure 4. Room temperature trajectories of measurement and training data with sample size $n = 256$, sampled uniformly over parameter space $\boldsymbol{\theta} \sim \mathcal{U}(\boldsymbol{\theta}_-, \boldsymbol{\theta}_+)$

.

Table 2. Optimization methods used in this paper for calibration. If the objective is scalar, a $MSE = \mu(x_{\text{sim}} - x_{\text{meas}})^2$ is used as objective, if the objective is residual, the vector of squared residuals at the simulation time steps $[(x_{\text{sim}}(t_0) - x_{\text{meas}}(t_0))^2, (x_{\text{sim}}(t_0 + \Delta t) - x_{\text{meas}}(t_0 + \Delta t))^2, \ldots]$ is used as objective. For all techniques from *SciPy*, default settings are used. Iterations are limited to reasonable values and tolerances are adapted to the *FMU*-settings.

Method Name	**Short Description**	**Objective**	**Gradient**	**Scope**
	Methods from SciPy (Virtanen et al. 2020)			
Powell	Conjugate direction method, sequentially performing one-dimensional optimization over an iteratively updated set of direction vectors	Scalar	No	Local
Nelder-Mead	Geometric operations (reflection, expansion, contraction, compression) on a simplex of points (Gao and Han 2012)	Scalar	No	Local
Sequential Least Squares Programming (SLSQP)	Iterative Method for nonlinear constrained optimization that integrates constraints by solving quadratic programming subproblems (Kraft 1988)	Scalar	Yes	Local
Least Squares Trust Region Reflective (LS-TRF)	Gradient-based algorithm, incorporating trust region strategies and reflective boundaries to improve convergence	Residuals	Yes	Local
Differential Evolution (Genetic Algorithm)	Population-based algorithm evolving a population of solution candidates with genetic operations (e.g., mutation)	Scalar	No	Global
	Methods introduced in this paper			
Multistart	first inital value $\boldsymbol{\theta}_{\text{init},0} = \boldsymbol{\theta}_{\text{init}}$, following initial values $j > 0 : \boldsymbol{\theta}_{\text{init,j}} \sim \boldsymbol{\mathcal{N}}(\boldsymbol{\mu} = \boldsymbol{\theta}_{\text{init}}, \boldsymbol{\sigma}^2 = 0.5(\boldsymbol{\theta}_+ - \boldsymbol{\theta}_-))$, repeated until a combination within the bounds $(\boldsymbol{\theta}_-, \boldsymbol{\theta}_+)$ is found.			
GA with PELS-VAE	PELS-VAE coupled with a genetic algorithm	Scalar	No	Global
GA with PELS-VAE + Polish	Phase 1: PELS-VAE coupled with genetic algorithm Phase 2: Polishing Result by LS-TRF with *FMU* of Modelica Model	Scalar	Yes (Phase 2)	Global (Phase 1)

neural-network (i.e. capable of evaluating 10 000 parameter combinations in a few seconds on a GPU) with a genetic algorithm to determine a parameter set that achieves global optimization. Additionally, we propose a 2-Phase approach in which the parameter combination determined by PELS-VAE coupled with a genetic algorithm serves as starting point for a polishing phase. The polishing phase employs the LS-TRF algorithm coupled with the *FMU* of the physical model to be calibrated. This approach is intended to compensate for inaccuracies that may arise when replacing the physical model with a machine learning surrogate model.

3 Results

3.1 Sensitivity Analysis

In order to evaluate the impact of different model parameters to the room temperature, we employ a sensitivity analysis based on Sobol indices as described in Appendix B. The result is shown in Figure 5, where for each time step the Sobol indices are plotted. Obviously the impact of different parameters changes with time: due to heating with TAC and air supply during daytime the "passive" building properties `UWin` and `rExt1` become less important.
This can be used in order to potentially limit the number of parameters in the overall analysis or in the training of the Autoencoder, as parameter dependencies with large impact are faster learned, that is less training is required (see section 3.2). Often this will be sufficient for the global phase 1 of the optimization approach described in this paper (see section 3.3.3).

3.2 Autoencoder Training

The Autoencoder training was carried out using different numbers of samples n, a varied dimension of the latent space ($\dim(\boldsymbol{z_x})$), and varied dimension of the hidden layers. The analysis, shown in Figure 6, was performed using the same test set ($n = 320$) for all experiments.

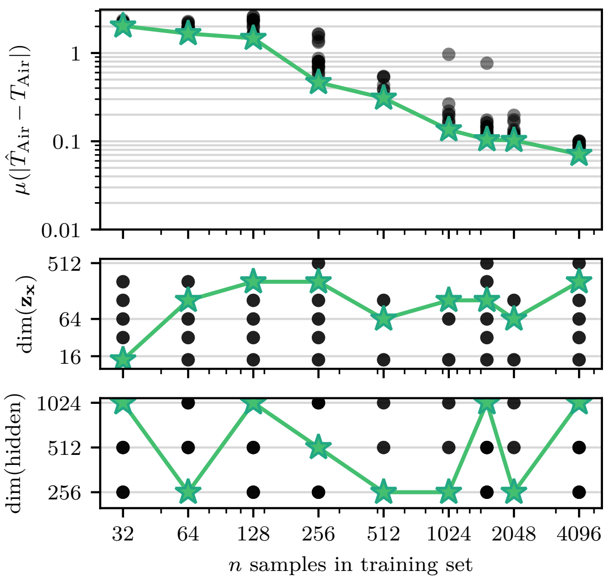


Figure 6. Hyperparameter variation (latent space dimension $\dim \boldsymbol{z_x}$ and dimension of hidden layers) over training sets with different number of samples n, tested with same uniformly sampled test set, all within $[\boldsymbol{\theta}_{-}, \boldsymbol{\theta}_{+}]$. The best-performing model for each training dataset size is marked by a star. (training performed for day 8-16 of identification timeframe)

Firstly, the Mean Absolute Error (MAE) was observed to decay with an increasing number of samples. Specifically, for 32 samples, the MAE was approximately 2, which decreased to around 0.07 for 4096 samples. Notably, with 1024 samples, the MAE reached 0.1, and further quadrupling the sample size only resulted in marginal improvements.

Secondly, models trained with different hyperpa-

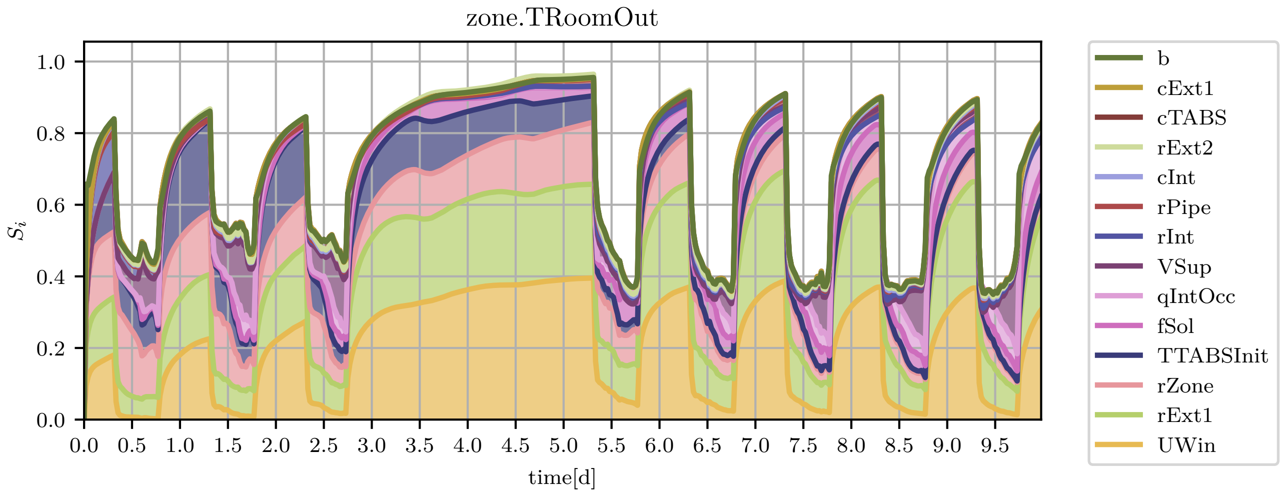


Figure 5. First order Sobol indices for model parameters, plotted ordered by mean value

rameters show variation in MAE. Although at higher numbers of samples the variations may be tolerable, at $n = 256$ the influence of hyperparameter selection lies in the range of 0.4 to 1.1 which might not be appropriate.

Lastly, the dimension of the latent space ($\dim(\boldsymbol{z_x})$) was found to scale with the complexity of the time series. Although the model had 14 parameters, the best latent space dimension are 64, 128, or 256. Reasons for this assumption are that $\boldsymbol{\theta}$ was also directly introduced in the decoder and we found by inspection, the worst performing models had low $\dim(\boldsymbol{z_x})$.

Overall, we find that the influence of chosen hyperparameters changes the training outcome, although its influence is limited, i.e. all hyperparameter sets produced results in comparable ranges with no "failing" hyperparameter sets. We conclude from this, that the Autoencoder training is quiet robust.

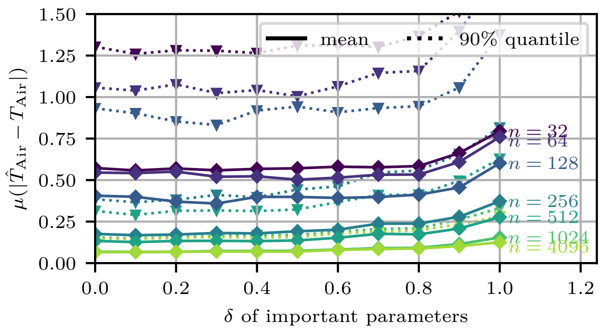
To further assess the performance of the Autoencoder, we have evaluated the prediction error using test sets with varying variability. In the case of large sample sizes n, the median lies at the midpoint of the parameter space of each dimension, approximated as $\tilde{\theta}_i \approx \frac{\theta_{i,+}+\theta_{i,-}}{2}$ with distance of $\Delta\theta_i = \frac{\theta_{i,+}-\theta_{i,-}}{2}$ to each bound. To generate the test sets, we sample with different δ as follows:

$$\boldsymbol{\theta} \sim \mathcal{U}(\tilde{\boldsymbol{\theta}} - \delta\boldsymbol{\Delta\theta}, \tilde{\boldsymbol{\theta}} + \delta\boldsymbol{\Delta\theta}), \quad \delta \in 0, 0.1, \ldots, 1.0$$

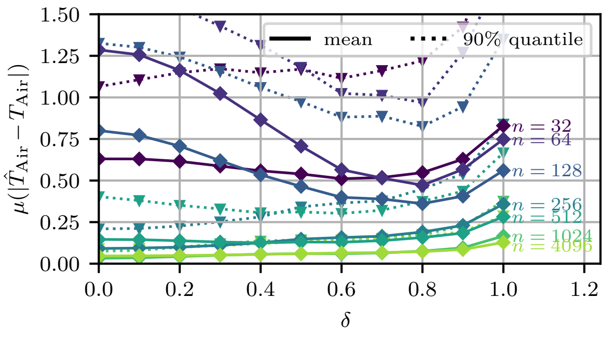
We generate two kinds of test sets: A category in that all parameters are varied and a category in that only the six most important parameters (subsection 3.1, Figure 5) are varied. The Mean Absolute Error (MAE) for these test sets, evaluated on the models trained with the best-performing hyperparameters determined previously, is presented in Figure 7. For the following, we can exclude a discussion about the numerical influence of the parameter value magnitudes as all parameters are normalized by mean and standard deviation before they are fed into the Neural Networks.

First, we take a closer look on the variation of important parameters (7a): We previously observed the MAE in Figure 6 with a variability of $\delta = 100\%$. However, by reducing the variability and excluding the border regions of the parameter space, the prediction error of the Autoencoder decreases. For instance, in the case of a training size of $n = 32$, the error is reduced from 0.8 Kelvin to approximately 0.6 Kelvin at 80% variability.

Figure 7 also includes the 90th percentile of the prediction error. It is evident that certain predictions exhibit considerably higher error than the mean prediction error, which can pose challenges in the optimization process, particularly with Autoencoder models trained on smaller datasets. However, by reducing the variability in the parameter space, the 90th percentile error also decreases.



(a) Only important parameters sampled with δ, for non-important parameters $\delta = 1$.



(b) All parameters randomly sampled.

Figure 7. Mean Absolute Error on randomly sampled test sets with different maximum deviations of parameter combinations from the median value of the training data ($\theta_i \sim \mathcal{U}(\tilde{\theta}_i - \delta\Delta\theta_i, \tilde{\theta}_i + \delta\Delta\theta_i)$). The mean absolute error over all time-series of a test set as well as the 90% quantile is given for the best performing models trained with different numbers of time series. (training performed for day 0-10 of identification timeframe)

Secondly, an analysis is conducted to examine the variation of all parameters, as shown in Figure 7b. Interestingly, it is observed that for small sampling sizes ($n = 32$ to 128), reducing the variability δ leads to an increase in the mean absolute error (MAE), while this trend does not persist for larger sampling sizes. This finding may initially seem counterintuitive, as one might expect that when varying all parameters, the MAE would decrease with overall less variability, compared to varying only the important ones and leaving the others unlimited. However, parameters that are considered less important contribute less to the observed variation in the output of the physical model. Consequently, when training sets are small, the Autoencoder faces challenges in capturing the influence of these less important parameters on the observed trajectories. By limiting the variation of all parameters to, for example, $\delta = 0.2$, a larger proportion of parameters resides in the inner part of parameter space, which can be more difficult for the Autoencoder to learn with small training sets as the majority of variance is produced by the influential parameters.

Consequently, this results in an increase in the mean absolute error.

From this analysis, we conclude that the Autoencoder performs better when predicting parameter combinations $\boldsymbol{\theta}$ that are more centered within the training parameter space. When selecting the bounds $[\boldsymbol{\theta}_-, \boldsymbol{\theta}_+]$, it should be ensured that they are larger than the parameter region where we anticipate the calibrated parameter results to lie. Furthermore, for small number of samples in the training set, the Autoencoder faces difficulties learning properly the influence of less influential parameters on the model output, while learning the impact of the more influential. However, to integrate the influence of the less influential parameters on the model output variation, they should still be sampled during training data generation. This property of the Autoencoder enables to resign on a Sensitivity Analysis before training it.

3.3 Model Calibration

In the following section, we present our findings regarding the curve-fitting methods minimizing MSE between model output and measurement outlined in Table 2. This section is organized as follows: firstly, we present the results obtained from the Optimizers directly coupled to the Modelica Model's FMU, along with the corresponding multistart approach (refer to Table 3), and gain insights to the uniqueness of a solution. Secondly, we showcase the optimization results achieved using the surrogate PELS-VAE Model (see Table 4) and highlight the advantages of our proposed method.

3.3.1 Direct Optimizer Coupling

First of all, one should keep in mind that the measurement signal is prone to error which results from measurement uncertainty of the temperature sensor ($\leq 0.5\,\mathrm{K}$ (Freund 2023)), the data processing and the position of the sensor in the room.
The calibration results obtained from the directly coupled optimizer are presented in Table 3. The majority of methods achieve a final Mean Squared Error (MSE) of approximately 0.01, although they vary significantly in terms of required model calls. Among the local optimizers, LS-TRF achieves the lowest number of iterations, with 261 model calls using the given initial value. SLSQP follows with 3-6 times higher iterations. The none-gradient optimizers Nelder-Mead and Powell perform less efficiently, requiring 5000 model calls (limited by the predefined iteration limit) with the given initial value. The notable difference between SLSQP with a scalar objective and LS-TRF with a vector-like/residual objective can be attributed to the fact that the residual objective allows after calculating the gradient for a more detailed consideration of the consequences of optimizer steps.

When initial values are poorly known, global opti-

Table 3. Calibration Results of Methods which were directly coupled with the Modelica Model's *FMU* with achieved $\mathrm{MSE}_f(\boldsymbol{\theta}_{opt})$.

Method	**model calls**	**MSE**
LS-TRF with initial guess	261	0.0117
SLSQP with initial guess	650	0.0149
LS-TRF with 16 starts	3486	0.0114
Nelder-Mead with initial guess	5000	0.0126
Powell with initial guess	5000	0.0881
LS-TRF with 32 starts	6311	0.0114
LS-TRF with 64 starts	12 612	0.0113
SLSQP with 16 starts	18 245	0.0117
Nelder-Mead with 16 starts	76 483	0.0117
Powell with 16 starts	79 120	0.0134
SLSQP with 64 starts	86 322	0.0114
Nelder-Mead with 64 starts	286 728	0.0110
Powell with 64 starts	31 9322	0.0125
Diff. Evolution with FMU	748 020	0.0104
Nelder-Mead with 256 starts	1 188 496	0.0111

mization strategies help to find the global minimum of a function. The Differential Evolution (Genetic) Algorithm uses unsurprisingly a high number model calls, namely 0.7 million. The introduced multistart-approach also quickly scale the number of required model calls, i.e. for the best performing algorithm LS-TRF 6311 calls with 32 different initial values.

3.3.2 Uniqueness of Solution

To gain more insights into the uniqueness of the solution to our optimization problem, a more detailed analyis of the best-performing algorithm LS-TRF was performed. To perform this a benchmark, a high number of starts (256) was chosen. The identified parameter combinations results were clustered with K-Means Clustering around common centroids (Pedregosa et al. 2011) with the 3 largest groups depicted in 8a, while the 5% best solutions are shown in 8b. From these results, we can infer two insights: First, the optimization problem is ambiguous: one parameter can compensate for the effect for another, e.g. in 8b, a high capacity `cExt1` of the external wall can compensate for low heat resistance `rExt2` and vice-versa. Furthermore, the optimization problem is clearly non-convex, i.e. it hast multiple local minima and the identified parameter combination is depending of the initial value when using local optimizer, which can be seen by the difference between the MSE of the best 5% results with 0.0114 and the average value of 0.0364. If the the problem would be convex, every initial value should lead to the same solution. Therefore, multiple parameter combinations might lead to equally well performing calibrated models.

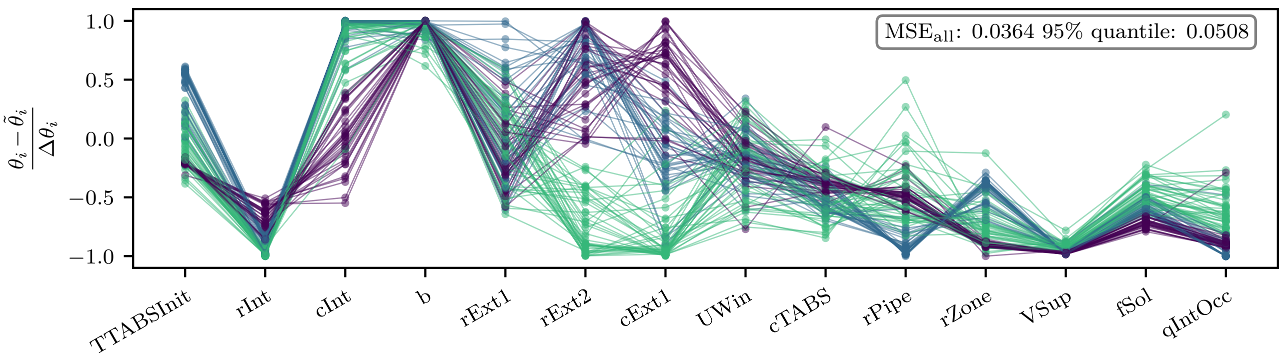
3.3.3 Calibration with Surrogate PELS-VAE Model

The results of the model calibration performed with the surrogate PELS-VAE Model are shown in Table 4 and Figure 9. For the MSE calculated based on the Autoencoder prediction ($\text{MSE}_{\boldsymbol{\phi}_{\text{de}},\boldsymbol{\varphi}_{\text{re}}}(\hat{\boldsymbol{\theta}}_{opt})$), MSE-values comparable to the direct coupling of Optimizer and Modelica-Model (≈ 0.01) are achieved. However, as shown in subsection 3.2, the Autoencoder is prone to prediction errors, i.e. for some parameter combinations, the predicted room temperature trajectories are more faulty than others. Because of this, the MSE of the parameter combination determined by the GA and the Autoencoder, denoted as $\hat{\boldsymbol{\theta}}_{\text{opt}}$, calculated with the Modelica-Model f, $\text{MSE}_f(\hat{\boldsymbol{\theta}}_{opt})$, can be considerably larger than the predicted $\text{MSE}_{\boldsymbol{\phi}_{\text{de}},\boldsymbol{\varphi}_{\text{re}}}(\hat{\boldsymbol{\theta}}_{opt})$. This effect occurs at low numbers of training samples and decreases with higher sampling n_{train}, i.e. a MSE-gap of 1.05 to 3.44 at $n_{\text{train}} = 32$ to 128 is reduced to a gap of 0.02 to 0.22 at $n_{\text{train}} = 512$ to 4096. Although this increase might be negligible at low magnitude, for the results obtained with low number of training samples it might not be appropriate.

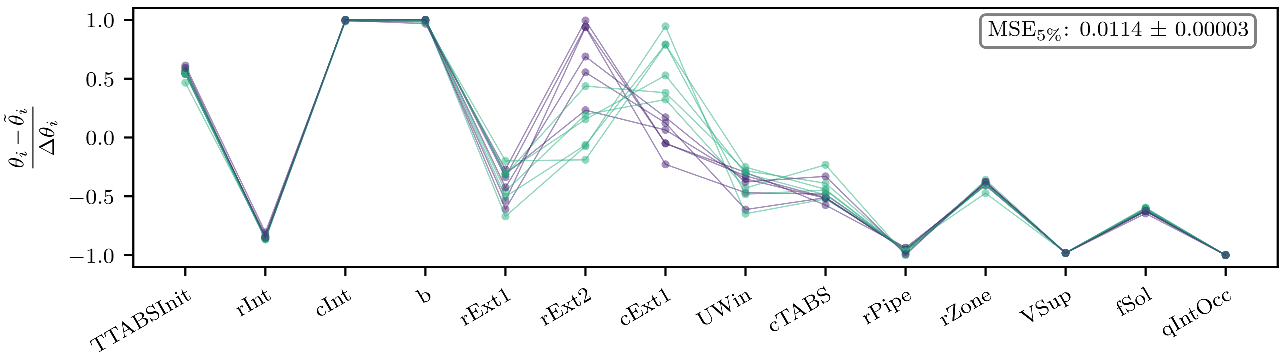
Table 4. Calibration Results of PELS-VA coupled with Differential Evolution Genetic Algorithm (GA) for different number of training samples n_{train}, MSE calculated by PELS-VAE ($\text{MSE}_{\boldsymbol{\phi}_{\text{de}},\boldsymbol{\varphi}_{\text{re}}}(\hat{\boldsymbol{\theta}}_{opt})$) and with FMU ($\text{MSE}_f(\hat{\boldsymbol{\theta}}_{opt})$) to determine prediction error introduced by the Autoencoder, number of steps n_{opt} of polishing with LS-TRF and achieved $\text{MSE}_f(\boldsymbol{\theta}_{opt})$.

n_{train}	MSE PELS-VAE + GA	MSE with FMU	n_{opt}	MSE after polish	n_{total}
32	0.0121	1.0623	228	0.0121	260
64	0.0128	1.6390	124	0.0128	188
128	0.0122	3.4509	171	0.0122	299
256	0.0117	0.7498	306	0.0117	562
512	0.0124	0.2486	139	0.0124	651
1024	0.0133	0.0816	65	0.0133	1089
1536	0.0128	0.0298	80	0.0128	1616
2048	0.0135	0.2080	65	0.0135	2113
4096	0.0118	0.0326	201	0.0118	4297

To compensate for that, polishing of the achieved results with the LS-TRF Algorithm, (local, gradient-based, vector-like objective), is performed. This process is illustrated in Figure 9. For all sampling sizes,



(a) 3 largest clusters of solutions, covering 110/256 results, clustered with `sklearn.cluster.KMeans` (Pedregosa et al. 2011) K-Means clustering around 10 centroids.



(b) Best 5% of solutions

Figure 8. Identified normalized parameters for Least-Squares Trust Region Reflective Algorithm with 256 starts.

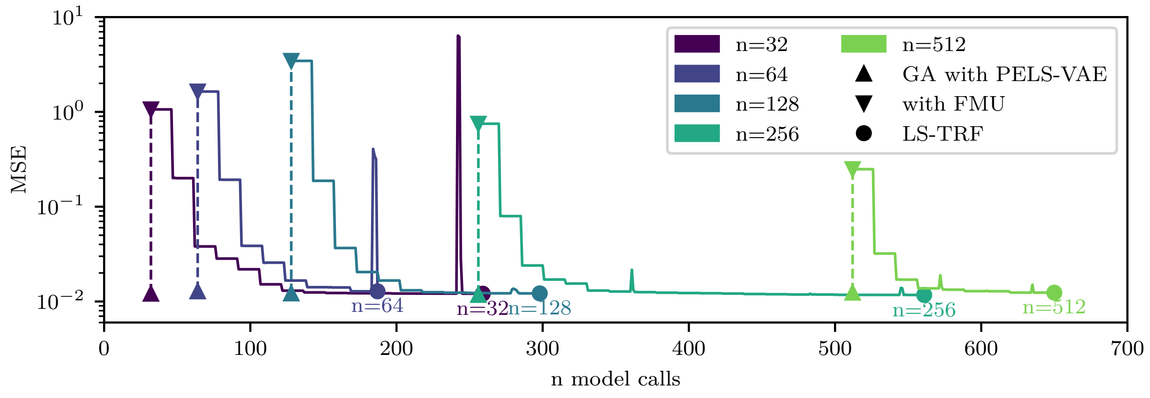


Figure 9. Calibration Results of PELS-VA coupled with Genetic Algorithm (GA), including the prediction gap and the MSE-trajectory during the LS-TRF Optimization.

the MSE is reduced considerable to a magnitude of $\text{MSE}_f(\boldsymbol{\theta}_{opt}) \approx 0.012$. More important, comparing the results for $n_{\text{train}} = 32, 64, 128$, a MSE comparable to that of the LS-TRF directly coupled with the Modelica model with an initial guess is achieved (Table 3), while requiring less or comparable model calls. This effect could be explained as following: As the Autoencoder learns the model reaction on different parameter combinations, especially for the most influential parameters (see subsection 3.2), it allows for a "screening" of parameter space to find a good starting point for the following gradient-based optimization with the exact Modelica-Model. At higher sampling sizes, the prediction gap decreases, which results in the number of model calls reduced as well.

Depending on the number of training samples, one might argue at which point we achieve a screening which is sufficient to call this approach a "global method".

To stress the advantage of this proposed novel method of model calibration: Using the Autoencoder allows a *screening of parameter space*, which *relieves* us of the *burden of finding an initial value* for the optimization, that could potentially even lead us into the "trap" of a local minimum.

4 Conclusion

In this paper, we address the challenges associated with physics based Modelica models increasing in complexity and computational expense regarding optimization-based multi parameter calibration. To overcome these issues, we present a novel approach that enables computationally efficient parameter calibration by using a Machine-Learning Surrogate.

To showcase our developed method, we use a simple thermal zone model implemented in Modelica, which allows to focus on the analysis of the proposes method.

The used Machine-Learning Surrogate is a Physics-Enhanced Latent Space Variational Autoencoder (PELS-VAE). It provides efficient model regularization and robust training. We propose to combine a PELS-VAE trained on a small dataset with a Genetic Algorithm (as PELS-VAE inference is computational cheap) to screen parameter space for well-performing parameter regions. To achieve best-performing results, we furthermore propose to polish the achieved result with a gradient-based residual-objective optimizer (LS-TRF).

To compare our approach to existing alternatives, we have tested a variety of optimizers and found significant variation in number of required model calls and strong dependence on initial values. When moving towards global optimization, the usage of multi-start approaches or global optimizer quickly scales significantly the number of model calls, making this potentially infeasible for computational demanding system models.

We were additionally able to show that the chosen optimization problem is non-convex and has ambiguous solutions.

We also perform a detailed analysis of the PELS-VAE application. By analyzing the training process, we find that hyperparameter variation has limited impact on the training process, i.e. we have a robust training, while predicting time-series that are more centered within the training parameter space exhibit considerably lower prediction error.

Our results provide evidence that even PELS-VAE trained with small datasets (32-128 samples) and resulting high prediction errors proved effective to screen parameter space for initial values which are then used in a gradient based optimizer. We provide indications that the PELS-VAE is able to capture the impact of most-influential parameters on small training sets. Comparing to the best-performing optimizer with the need for an initial value, we were able to show that our initial value free method achieved comparable MSE with comparable number

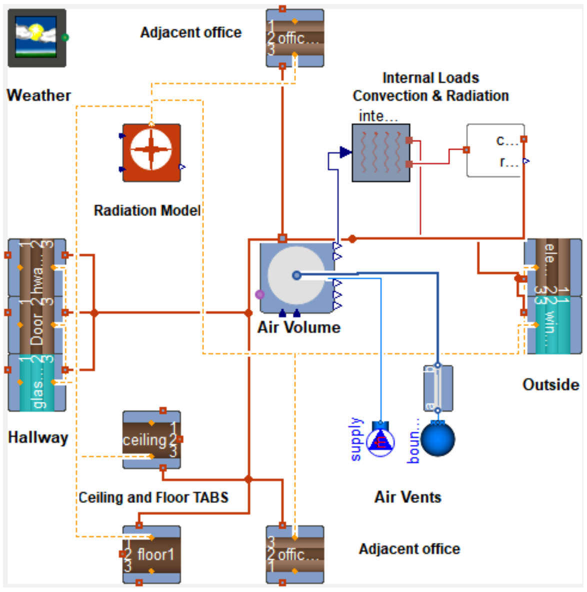


Figure 10. Physics based model of the office with XRG-simulation's HumanComfort Library

of model calls.
In summary, our proposed method offers an effective solution for calibrating complex models. Using the PELS-VAE models allows for a screening of parameter space with a low number of model calls, and relieves us from the burden of fining suitable initial values for local optimizers.
For future work, our method will be applied to other examples like a White-Box Model of the office (see Figure 10) to prove its suitability for various kind of optimization problems. Furthermore, the training process could be improved by adaptive online data generation, narrower parameter ranges, other layers in the network and embedding of multiple Modelica model outputs.

Acknowledgements

A major part of this work has been carried out within the the PHyMoS research project (Proper Hybrid Models for Smarter Vehicles) funded by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) under grant number 19I20022F. J.A. and J.B. thank Yi Zhang and Lars Mikelsons for numerous insightful discussions and support with code examples for implementing PELS-VAE. Moreover J.A. and J.B. thank the XRG team for support. We also thank the anonymous referees for valuable remarks.

Funded by the European Union
NextGenerationEU
Supported by:
Federal Ministry for Economic Affairs and Climate Action
on the basis of a decision by the German Bundestag

References

Arendt, Krzysztof et al. (2018). "ModestPy: an open-source python tool for parameter estimation in functional mock-up units". In: *Proceedings of the 1st American Modelica Conference*. Modelica Association and Linköping University Electronic Press, pp. 121–130.
Bhosekar, Atharv and Marianthi Ierapetritou (2018). "Advances in surrogate based modeling, feasibility analysis, and optimization: A review". In: *Computers & Chemical Engineering* 108, pp. 250–267. DOI: https://doi.org/10.1016/j.compchemeng.2017.09.017.
Burgess, Christopher P. et al. (2018). *Understanding disentangling in β-VAE*. arXiv: 1804.03599 [stat.ML].
Duus, Kristian and Gerhard Schmitz (2021). "Experimental investigation of sustainable and energy efficient management of a geothermal field as a heat source and heat sink for a large office building". In: *Energy and Buildings* 235, p. 110726. DOI: 10.1016/j.enbuild.2021.110726.
FMPy (2023-06-10). *FMPy, A free Python library to simulate Functional Mock-up Units (FMUs)*. URL: https://github.com/CATIA-Systems/FMPy (visited on 2023-06-10).
Freund, Svenne (2023-03). "Modellbasierte Prädiktive Regelung komplexer gebäudetechnischer Anlagen zur Optimierung der Energieeffizienz und des Komforts". PhD thesis. DOI: 10.15480/882.5018.
Freund, Svenne and Gerhard Schmitz (2021-03). "Implementation of model predictive control in a large-sized, low-energy office building". In: *Building and Environment* 197, p. 107830. DOI: 10.1016/j.buildenv.2021.107830.
Gao, Fuchang and Lixing Han (2012-05). "Implementing the Nelder-Mead simplex algorithm with adaptive parameters". In: *Computational Optimization and Applications* 51, pp. 259–277. DOI: 10.1007/s10589-010-9329-3.
Hart, Joey (2018). "Sobol' Indices for Sensitivity Analysis with Dependent Inputs ". In: *SIAM Conference on Uncertainty Quantification, Garden Grove, California, USA*. fetched June, 10th 2023. URL: https://www.pathlms.com/siam/courses/7376#.
Jordan, Jeremy (2018a-03-18). *Introduction to autoencoders*. URL: https://www.jeremyjordan.me/autoencoders/ (visited on 2023-06-10).
Jordan, Jeremy (2018b-03-19). *Variational autoencoders*. URL: https://www.jeremyjordan.me/variational-autoencoders/ (visited on 2023-06-10).
Kathirgamanathan, Anjukan et al. (2021). "Data-driven predictive control for unlocking building energy flexibility: A review". In: *Renewable and Sustainable Energy Reviews* 135, p. 110120. DOI: 10.1016/j.rser.2020.110120.
Kingma, Diederik P and Max Welling (2022). *Auto-Encoding Variational Bayes*. arXiv: 1312.6114 [stat.ML].
Koschenz, M. and B. Lehman (2000). "Thermoaktive Bauteilsysteme tabs". In: *EMPA Energiesysteme/Haustechnik*. 1st ed. Dübendorf.
Kraft, Dieter (1988). "A software package for sequential quadratic programming". In: *Forschungsbericht-Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*.

Martínez-Palomera, Jorge, Joshua S. Bloom, and Ellianna S. Abrahams (2020). *Deep Generative Modeling of Periodic Variable Stars Using Physical Parameters.* arXiv: 2005.07773 [astro-ph.IM].

Martinez-Viol, Victor et al. (2022). "Automatic model calibration for coupled HVAC and building dynamics using Modelica and Bayesian optimization". In: *Building and Environment* 226, p. 109693. ISSN: 0360-1323. DOI: https://doi.org/10.1016/j.buildenv.2022.109693.

Murphy, Kevin P. (2022). *Probabilistic Machine Learning: An introduction.* MIT Press. URL: probml.ai.

Murphy, Kevin P. (2023). *Probabilistic Machine Learning: Advanced Topics.* MIT Press. URL: http://probml.github.io/book2.

Niemann, Peter and Gerhard Schmitz (2020). "Impacts of occupancy on energy demand and thermal comfort for a large-sized administration building". In: *Building and environment* 182, pp. 1–15. DOI: 10.15480/882.2857.

Odaibo, Stephen (2019). *Tutorial: Deriving the Standard Variational Autoencoder (VAE) Loss Function.* arXiv: 1907.08956 [cs.LG].

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

SALib (2023-06-10). *SALib, Sensitivity analysis library for systems modeling.* URL: https://salib.readthedocs.io/en/latest/ (visited on 2023-06-10).

Saltelli, A. et al. (2008). *Global Sensitivity Analysis: The Primer.* Chichester (England): Wiley. ISBN: 978-0-470-05997-5.

Shahriari, Bobak et al. (2016). "Taking the Human Out of the Loop: A Review of Bayesian Optimization". In: *Proceedings of the IEEE* 104.1, pp. 148–175. DOI: 10.1109/JPROC.2015.2494218.

Sobol, I.M. (1993). "Sensitivity Estimates for Nonlinear Mathematical Models". In: *Mathematical Modelling and Computational Experiments* 4, pp. 407–414.

Sourbron, M (2012). "Dynamic thermal behaviour of buildings with concrete core activation". Dissertation. Katholieke Universiteit Leuven.

University of California, The Regents of the (2023-08-08). *GenOpt Generic Optimization Program.* URL: https://simulationresearch.lbl.gov/GO/index.html (visited on 2023-08-08).

Virtanen, Pauli et al. (2020). "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17, pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

Wüllhorst, Fabian et al. (2022). "AixCaliBuHA: Automated calibration of building and HVAC systems". In: *Journal of Open Source Software* 7.72, p. 3861. DOI: 10.21105/joss.03861.

XRG Simulation GmbH (2023-08-08). *ModelOpt.* URL: http://xrg-simulation.de/de/produkte/applications/modelopt (visited on 2023-08-08).

Zhang, Yi and Lars Mikelsons (2022-07). "Sensitivity-Guided Iterative Parameter Identification and Data Generation with BayesFlow and PELS-VAE for Model Calibration". In: DOI: 10.21203/rs.3.rs-1898389/v1.

A Physics Enhanced Latent Space Variational Autoencoder (PELS-VAE)

This is a explanation with more detail but simplifications intended for the Modelica Community to gain understanding. For theory without simplifications, please refer to (Kingma and Welling 2022), (Murphy 2022) and (Murphy 2023).

Machine Learning can be done with a probabilistic perspective, such that the quantities of interests are modeled as random variables (Murphy 2022). In stochastic variational inference, it is assumed that the data $\boldsymbol{x}$ to be learned has initially emerged from a latent variable distribution $p(\boldsymbol{z})$ with a conditional probability density function $p(\boldsymbol{x}|\boldsymbol{z})$ (Kingma and Welling 2022).

To model this process within means of unsupervised learning, we have to infer the stochastic latent variable $\boldsymbol{z}$ by a recognition model $p_*(\boldsymbol{z}|\boldsymbol{x})$ from a data sample $\boldsymbol{x}$ and then reconstruct the data with a generation model $p_*(\boldsymbol{x}|\boldsymbol{z})$. As both the true recognition and generation model are inaccessible to us, we model them by using Neural Networks; $q_{\psi_{\text{en}}}(\boldsymbol{z}|\boldsymbol{x})$ for the recognition model and $p_{\phi_{\text{de}}}(\boldsymbol{x}|\boldsymbol{z})$ for the generation model.

For training probabilistic models, one commonly tries to maximize the marginal likelihood of the data, $\sum_{i=0}^{N}\log p(\boldsymbol{x}_i)$. This is the likelihood the network structure assigns to the probability density function at $\boldsymbol{x}_i$, if $\boldsymbol{x}_i$ was inserted. One can think of this as following: If the probability of a data sample $\boldsymbol{x}_i$ is high, the information content it carries is low, i.e. the characteristics are learned by the probability distributions which are modeled by the Neural Networks ψ_{en} and ϕ_{de}. (Odaibo 2019).

One can show (Odaibo 2019) that the right hand side of

$$\log p(\boldsymbol{x}_i) \geq -\mathbb{KL}(q_{\psi_{\text{en}}}(\boldsymbol{z}|\boldsymbol{x}_i) \| p(\boldsymbol{z})) + \mathbb{E}_{\boldsymbol{z}\sim q_{\psi_{\text{en}}(\boldsymbol{z}|\boldsymbol{x}_i)}}(\log p_{\phi_{de}}(\boldsymbol{x}_i|\boldsymbol{z})) = \mathcal{G} \quad (3)$$

is a lower bound (namely the evidence lower bound $\mathcal{G}$, ELBO) to the likelihood of a data sample $\log p(\boldsymbol{x}_i)$, which we seek to maximize. In general, the used network structure is a deterministic one, i.e. $y = f(x)$. To use the network for the approximation of probablistic distributions, two tricks are applied. First, the encoder model ψ_{en} is used to predict the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}$ for a data sample $\boldsymbol{x}_i$ of the distribution $p(\boldsymbol{z})$, which is prescribed to be a multivariate gaussian (i.e. normal) distribution, i.e. $\boldsymbol{z} \sim \boldsymbol{\mathcal{N}}(\boldsymbol{\mu}, \boldsymbol{\sigma})$. Assuming this, an analytical expression for the Kullback-Leibler-Divergence-Term ($\mathbb{KL}$) in Equation 3 can be found (Odaibo 2019). Then, sampling of a random, normal distributed auxiliary variable $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \mathbb{I})$ is required to obtain samples $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$, where $\odot$ represent element-wise multiplication. Using this, an estimate for the second term in Equation 3, the reconstruction likelihood, can be found. The reconstruction term can be determined from the network structure (Kingma and Welling 2022), however, in our model, we follow the approach of (Martínez-Palomera, Bloom, and Abrahams 2020) and use the negative mean squared error between prediction and ground truth, $\mathrm{MSE}(\hat{\boldsymbol{x}}, \boldsymbol{x})$ as representation of the reconstruction likelihood. Taking this together, the evidence lower bound for our network becomes for a training sample $\boldsymbol{x}_i$ with L samples in latent space

$$\mathcal{G} \approx \sum_{j=1}^{\dim(\boldsymbol{z})} \frac{1}{2}\Big[1 + \log(\sigma_j^2) - \sigma_j^2 - \mu_j^2\Big] - \beta \ \mathrm{MSE}\Big(\frac{1}{L}\sum_{l=0}^{L} \hat{\boldsymbol{x}}_{i,l}, \boldsymbol{x}_i\Big) \quad (4)$$

which should be maximized. When minimizing in a Optimizer, we should do this with $\mathcal{L} = -\mathcal{G}$. Furthermore, the hyperparameter β is added to help disentangling the latent space distribution $\boldsymbol{z}$ (Burgess et al. 2018). Finally (Martínez-Palomera, Bloom, and Abrahams 2020) introduce the physical parameters $\boldsymbol{\theta}$ as inputs for all sub-models.

To use the Variational Autoencoder as a generative model, the representation of the Modelica model parameters $\boldsymbol{\theta}$ within the latent space must be traceable, which is why (Zhang and Mikelsons 2022) added a regression model in a Teacher-Student Architecture. The regression model tracks $\boldsymbol{\theta}$ in latent space, i.e. $\boldsymbol{\mu}_{\mathrm{re}}, \boldsymbol{\sigma}_{\mathrm{re}} = \varphi_{\mathrm{re}}(\boldsymbol{\theta})$. MSE-losses of this model are added to the loss-function to train all models simultaneously. Further details of the implementation can be found in (Zhang and Mikelsons 2022). The overall objective function becomes with this for a batch size N

$$(\psi_{\mathrm{en}}, \phi_{\mathrm{de}}, \varphi_{\mathrm{re}}) = \operatorname{argmin} \sum^{N} \ \mathcal{L} \ + \ \mathrm{MSE}(\boldsymbol{\mu}, \boldsymbol{\mu}_{re}) + \ \mathrm{MSE}(\boldsymbol{\sigma}, \boldsymbol{\sigma}_{re}) \quad (5)$$

Finally, a well-trained PELS-VAE can replace the physical model by determining the latent space representation of the physical parameters with the regressor model (6): the time-series $\boldsymbol{x}$ is reconstructed after sampling multiple times $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$ by the decoder as the mean of the outputs (7).

$$\{\boldsymbol{\mu}_{\mathrm{re}}, \boldsymbol{\sigma}_{\mathrm{re}}\} = \varphi_{\mathrm{re}}(\boldsymbol{\theta}) \quad (6)$$

$$\hat{\boldsymbol{x}} = \frac{1}{L}\sum^{L} \phi_{de}(\boldsymbol{z}) \quad (7)$$

B Global Sensitivity Analysis

In the following we sketch the idea of Sobol indices. For a more elaborate and mathematical sound introduction we refer to Hart (2018) or the book by Saltelli et al. (2008). A comprehensive implementation of the required functions is available in SALib (2023).

Consider X to be a real continuos random variable with probability density function $p_X(x)$, such that $\int_{-\infty}^{\infty} p_X(x)dx = 1$. X can be thought of as a specific measurement setup, giving a measured value x each time the experiment is carried out. If a large number of experiments is conducted and the obtained results x are collected into a histogram, then this histogram resembles the probability density distribution $p_X(x)$, that characterizes the experiment X. The probability P of measuring x inside the interval $[T_1, T_2]$ is $P(x \in [T_1, T_2]) = \int_{T_1}^{T_2} p_X(x)dx$. We define the **mean** μ_X of x as $\mu_X = \mathbb{E}_{p_X}[X] = \int_{-\infty}^{\infty} x \cdot p_X(x)dx$. The **variance** of x is defined as $\mathrm{Var}(X) = \mathbb{E}_{\mathbb{I}_{\mathbb{X}}}[(X - \mathbb{E}(X))^2] = \int_{-\infty}^{\infty}(x - \mu_X)^2 \cdot p_X(x)dx$. For a function $f(X)$ of the random variable X one can equivalently define its mean value $\mu_F = \mathbb{E}_{p_X}[f(X)] = \int_{-\infty}^{\infty} f(x) \cdot p_X(x)dx$ and its variance $\mathrm{Var}_{p_X}(F) = \int_{-\infty}^{\infty}(f(x) - \mu_F)^2 \cdot p_X(x)dx$.

Now assume that a system model is evaluated on a fixed scenario (=fixed time series of boundary conditions) for different variations of its model parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_r)$. If we look at a specific output y_t of the model at a specific timestep t then we can interpret our model as a map $f : \boldsymbol{\theta} \to y_t(\boldsymbol{\theta})$. Now assume for a moment that the parameters $\boldsymbol{\theta}$ are an r-dimensional random variable Θ with known probability density function $p_{\boldsymbol{\Theta}}$ and f to be square integrable. Then it can be shown (Sobol 1993) that

$$\begin{aligned} f(\boldsymbol{\theta}) &= f_0 + \sum_{i=1}^{r} f_i(\theta_i) + \sum_{1 \le i < j \le r} f_{i,j}(\theta_i, \theta_j) \\ &\quad + \ldots + f_{1,2,\ldots,r}(\theta_1, \theta_2, \ldots, \theta_r) \\ &= f_0 + \sum_{k=1}^{r} \sum_{|\boldsymbol{u}|=k} f_{\boldsymbol{u}}(\boldsymbol{\theta}_{\boldsymbol{u}}) \end{aligned} \quad (8)$$

where in the last line we summarize the previous line by the sum over the multi label $\boldsymbol{u}$ representing all possible subsets $\boldsymbol{u} \subseteq \{\theta_1, \ldots, \theta_r\}$ having $|\boldsymbol{u}| = k$ elements. Moreover we have the special expectation value functions

$$\begin{aligned} f_0 &= \mathbb{E}_{p_{\boldsymbol{\Theta}}}[f(\boldsymbol{\theta})] \\ f_i(\theta_i) &= \mathbb{E}_{p_{\boldsymbol{\Theta}}}[f(\boldsymbol{\theta})|\theta_i] - f_0 \\ f_{i,j}(\theta_i, \theta_j) &= \mathbb{E}_{p_{\boldsymbol{\Theta}}}[f(\boldsymbol{\theta})|\theta_i, \theta_j] - f_i(\theta_i) - f_j(\theta_j) - f_0 \\ &\vdots \\ f_{\boldsymbol{u}}(\boldsymbol{\theta}_{\boldsymbol{u}}) &= \mathbb{E}_{p_{\boldsymbol{\Theta}}}[f(\boldsymbol{\theta})|\boldsymbol{u}] - \sum_{k=1}^{|\boldsymbol{u}|-1} \sum_{\substack{\boldsymbol{v} \subset \boldsymbol{u} \\ |\boldsymbol{v}|=k}} f_{\boldsymbol{v}}(\boldsymbol{\theta}_{\boldsymbol{v}}) - f_0 \end{aligned}$$

with the conditional expectation values $\mathbb{E}_{p_{\boldsymbol{\Theta}}}[f(\boldsymbol{\theta})|\boldsymbol{u}] = \int_{-\infty}^{\infty} f(\boldsymbol{\theta}) p_{\boldsymbol{\Theta}}(\boldsymbol{\theta}) d\boldsymbol{\theta}_{\boldsymbol{\theta} \setminus \boldsymbol{u}}$, where $\boldsymbol{\theta} \setminus \boldsymbol{u}$

is the complement of $\boldsymbol{u} \subseteq \{\theta_1, \ldots, \theta_r\}$. Now if all elements $\{\theta_1, \ldots, \theta_r\}$ are statistically independent then it can be shown (Sobol (1993)) that

$$\operatorname{Var}(f(\boldsymbol{\theta})) = \sum_{k=1}^{r} \sum_{|\boldsymbol{u}|=k} \operatorname{Var}\left(f_{\boldsymbol{u}}(\boldsymbol{\theta}_{\boldsymbol{u}})\right) \tag{9}$$

that is the overall variance $\operatorname{Var}(f(\boldsymbol{\theta})) = \mathbb{E}_{p_{\boldsymbol{\Theta}}}\left[(f(\boldsymbol{\theta}) - f_0)^2\right]$ is the sum of all variances of the subset functions $f_{\boldsymbol{u}}$. Then the Sobol index $S_{\boldsymbol{u}}$ of the subset $\boldsymbol{u} \subseteq \{\theta_1, \ldots, \theta_r\}$ measures the relative contribution of $\boldsymbol{\theta}_{\boldsymbol{u}}$ to the total variance of $f(\boldsymbol{\theta})$:

$$S_{\boldsymbol{u}} = \frac{\operatorname{Var}\left(f_{\boldsymbol{u}}(\boldsymbol{\theta}_{\boldsymbol{u}})\right)}{\operatorname{Var}(f(\boldsymbol{\theta}))} \tag{10}$$

Moreover the total Sobol index $T_{\boldsymbol{u}}$ measures the relative contribution of all members of $\boldsymbol{u}$ to the total variance of $f(\boldsymbol{\theta})$:

$$T_{\boldsymbol{u}} = \sum_{\boldsymbol{v} \cap \boldsymbol{u} \neq \emptyset} S_{\boldsymbol{v}} \tag{11}$$

Finally the first order Sobol indices are those $S_{\boldsymbol{u}}, T_{\boldsymbol{u}}$ which are defined on single element subsets $|\boldsymbol{u}| = 1$, that is $\boldsymbol{u} = \{\theta_1, \{\theta_2\}, \ldots, \{\theta_r\}\} = \{\theta_k\}$. Then the Sobol indices S_k, T_k measure the importance or sensitivity of $\operatorname{Var}(f(\boldsymbol{\theta}))$ to $\{\theta_k\}$:

- the first order Sobol index S_k measures the contribution of θ_k
- the total Sobol index T_k measures the contribution of all interactions involving θ_k

The previous discussion is valid for a specific model output y_t at a single time step t so far. The extension to time series $\{t_1, t_2, \ldots, t_n\}$ is straight forward: one simply computes the Sobol indices for all parameters at each time step. In this way one can observe the impact of different model parameters to the chosen output y at several times. This is especially important for transient scenarios, as the sensitivity of the model output to the values of a specific model parameter may vary over time, as can be seen in Figure 5.